ELSEVIER

Artificial Intelligence 112 (1999) 213–232

Artificial Intelligence

www.elsevier.com/locate/artint

Research Note

Towards a characterisation of the behaviour of stochastic local search algorithms for SAT

Holger H. Hoos [a,*], Thomas Stützle [b,1]

[a] *University of British Columbia, Department of Computer Science, 2366 Main Mall, Vancouver, British Columbia, Canada V6T 1Z4*

[b] *FB Informatik, FG Intellektik, TU Darmstadt, Alexanderstr. 10, D-64283 Darmstadt, Germany*

Received 24 June 1998; received in revised form 27 May 1999

Abstract

Stochastic local search (SLS) algorithms have been successfully applied to hard combinatorial problems from different domains. Due to their inherent randomness, the run-time behaviour of these algorithms is characterised by a random variable. The detailed knowledge of the run-time distribution provides important information about the behaviour of SLS algorithms. In this paper we investigate the empirical run-time distributions for WalkSAT, one of the most powerful SLS algorithms for the Propositional Satisfiability Problem (SAT). Using statistical analysis techniques, we show that on hard Random-3-SAT problems, WalkSAT's run-time behaviour can be characterised by exponential distributions. This characterisation can be generalised to various SLS algorithms for SAT and to encoded problems from other domains. This result also has a number of consequences which are of theoretical as well as practical interest. One of these is the fact that these algorithms can be easily parallelised such that optimal speedup is achieved for hard problem instances.

Keywords: SAT; Stochastic local search; Empirical evaluation of algorithms; Run-time distributions; Parallelisation

1. Introduction

Recent successes in using stochastic local search (SLS) algorithms to practically solve hard combinatorial problems from various domains have stirred considerable interest and inspired a quickly growing body of research. Based on earlier SLS approaches [8,27] to solving the satisfiability problem in propositional logic (SAT), current SLS algorithms

* Corresponding author. Email: hoos@cs.ubc.ca.

[1] Email: stuetzle@informatik.tu-darmstadt.de.

0004-3702/99/$ – see front matter
PII: S0004-3702(99)00048-X

like HSAT [5], the Breakout method [24], WalkSAT [26], or Novelty [21] do not only outperform state-of-the-art systematic search methods for a variety of problem classes, but have also shown to be competitive with the best algorithms on domains like planning [19] or network routing [18].

Because the theoretical understanding of the behaviour of stochastic local search algorithms is still very limited, most of the work in this field is based on empirical methods and computational experiments. The kind of conclusions that may be drawn from their empirical analysis depends strongly on the empirical methodology applied. In particular, the empirical methodology has to take into account two important sources of randomness in the performance of an SLS algorithm. One is that for a given problem instance, the run-time needed to find a solution varies as a consequence of the inherent randomness of SLS algorithms, caused by randomised decisions like random initial solutions or randomly biased moves. The other is that the solution cost for solving a given problem instance depends on the instance itself, which (in cases like Random-3-SAT) might be randomly generated.

In this work we clearly separate these sources of randomness. The methodology we use is based on analysing the run-time behaviour of SLS algorithms on single instances drawn from a randomised problem class. Based on the empirical run-time distributions observed for a particular SLS algorithm applied to individual problem instances, we formulate a general hypothesis regarding the type of these run-time distributions for the given algorithm and problem class. This hypothesis is tested and validated by running a series of experiments on a large number of instances sampled from the given random problem distribution. Here, we use this generic methodological approach to analyse the performance of WalkSAT applied to Random-3-SAT, a prominent randomised NP-complete subclass of SAT consisting of propositional formulae in conjunctive normal form with exactly three literals per clause. Specifically we show that when applied to hard random 3-SAT instances, *WalkSAT's run-time behaviour can be characterised by exponential distributions.* This result also holds for various SAT-encoded problem classes, such as Blocks World Planning and Graph Colouring, and some recently proposed improved variants of the original WalkSAT algorithm. It has a number of significant implications, the most interesting of which might be the fact that SLS algorithms displaying this type of behaviour can be easily parallelised with optimal speedup.

The paper is structured in the following way. We begin with reviewing basic notions concerning SLS algorithms for SAT and introducing our empirical methodology. Then we present empirical analyses of the behaviour of several SLS algorithms when applied to randomised problem distributions and encoded problem instances from other domains which lead to our main result. Next, we discuss several interesting implications of this result. After comparing our approach with related work, we conclude with a brief summary of the main contributions of this work and pose some questions indicating possible directions for further research.

2. SLS algorithms for SAT

Local search is a widely used, general approach for solving hard combinatorial search problems. Stochastic local search can be interpreted as performing a biased random walk

```
procedure LocalSearch
    input CNF formula Φ, maxTries, maxSteps
    output satisfying assignment of Φ or "no solution found"
    for i := 1 to maxTries do
        s := random truth assignment;
        for j := 1 to maxSteps do
            if s satisfies Φ then return s;
            else
                x := chooseVariable(s, Φ);
                s := s with truth value of x flipped;
            end if
        end for
    end for
    return "no solution found";
end Local Search;
```

Fig. 1. Outline of a general local search procedure for SAT; actual SLS algorithms differ mainly in the variable selection function *chooseVariable*(s, Φ).

in the search space defined by the given problem instance; for SAT, the search space is the set of all possible truth assignments of the variables appearing in the given propositional formula in conjunctive normal form (CNF). A CNF formula Φ over n truth variables $x_1, x_2, \ldots, x_n$ (with domain {*true*, *false*} each), is a conjunction of m clauses $c_1, c_2, \ldots, c_m$. Each clause c_i is a disjunction of one or more literals, where a literal l_j is a variable x_j or its negation $\neg x_j$, i.e., $c_i = \bigvee_{j=1}^{m_i} l_j$. A formula is satisfiable, if an assignment of truth values to all variables can be found which simultaneously satisfies all clauses; otherwise the formula is unsatisfiable.

A general outline of an SLS algorithm for SAT is given in Fig. 1. The generic procedure starts with some truth assignment, which is randomly chosen from the set of all possible assignments according to a uniform distribution. Then it tries to reduce the number of violated clauses by iteratively flipping some variable's truth value, where the selection of the variable depends on the formula Φ and the current assignment s. If after a maximum of *maxSteps* flips no solution is found, the algorithm restarts from a new random initial assignment. If after a given number *maxTries* of such tries still no solution is found, the algorithm terminates unsuccessfully.

In this article we concentrate on SLS algorithms based on the WalkSAT architecture [26], one of the best performing local search approaches for SAT [21]. This architecture is based on a two-stage variable selection process focused on the variables occurring in currently unsatisfied clauses. For each local search step, in a first stage a currently unsatisfied clause c is randomly selected. In a second step, one of the variables appearing in c is then selected according to some heuristic h, biased to increase the total number of satisfied clauses; this variable is then flipped to obtain the new assignment.

For the original WalkSAT algorithm, in the following referred to simply as WalkSAT, a pseudocode of the function *chooseVariable*(s, Φ) is given in Fig. 2. Here, the function *randomPick* randomly selects an element of a set T according to a uniform distribution

```
function chooseVariable-WalkSAT(s, Φ)
    c := randomPick({c | c clause of Φ and c not satisfied under s});
    V := {x | x appears in c and break(Φ, s, x) is minimal};
    v := randomPick(V);
    if break(Φ, s, v) = 0 then
        return v;
    else
        with probability wp: v' := randomPick({x | x appears in c});
        otherwise v' := v;
        return v';
    end if
end chooseVariable-WalkSAT;
```

Fig. 2. Outline of the function *chooseVariable* for WalkSAT.

(i.e., each $t \in T$ is chosen with a probability $1/|T|$, where $|T|$ is the number of elements in set T).[2] For WalkSAT, the heuristic h chooses a variable with a minimal value $break(\Phi, s, x)$ from the selected clause, where $break(\Phi, s, x)$ is the number of clauses which are satisfied by the current assignment s but would become unsatisfied if variable x were flipped. If in the selected clause variables can be flipped without violating other clauses (i.e., $\exists x\colon break(\Phi, s, x) = 0$), one of these is randomly chosen. Otherwise, with a fixed probability *wp* a variable is randomly chosen (according to a uniform distribution) from the clause and with probability $1 - wp$ one of the variables minimising the number of breaks is picked. The walk probability *wp* (also called noise setting) is the most important parameter influencing the algorithms' overall performance.

In this article we also present computational results with two more recent WalkSAT variants. In WalkSAT with tabu-search (WalkSAT/Tabu) [21], heuristic h always picks one of the variables minimising the number of breaks. However, variables which have been flipped during the last *tl* iterations, where *tl* is a parameter called *tabu list length*, are declared tabu and are not considered for flipping. To realise this, every variable x has associated an age, $age(x)$, which is defined as the number of local search steps (i.e., variable flips) since x has been flipped the last time.[3] If all variables in the chosen clause are tabu, no variable is flipped (a so called *null-flip*). In Fig. 3 we give the pseudocode for WalkSAT/Tabu. The second strategy is called Novelty [21]. This strategy chooses variables which when flipped lead to a maximal decrease in the total number of unsatisfied clauses ($score(x) = fix(\Phi, s, x) - break(\Phi, s, x)$, where $fix(\Phi, s, x)$ is the number of clauses which are not satisfied under assignment s but would become satisfied if the variable x were flipped). Ties are broken in favour of the least recently flipped variable. If the variable

[2] For the remainder of this paper, all random selections are considered to be made using uniform probability distributions if not explicitly indicated otherwise.

[3] When implementing this algorithm, instead of explicitly storing and updating the variable ages, it is more efficient to store for each variable the number of the iteration in which it was flipped last and to compare these numbers instead of the actual ages.

```
function chooseVariable-WalkSAT/Tabu(s, Φ)
    c := randomPick({c | c clause of Φ and c not satisfied under s});
    V := {x | x appears in c and break(Φ, s, x) is minimal and age(x) ⩾ tl};
    if V ≠ {} then
        v' := randomPick(V);
        return v';
    else
        return {};
    end if
end chooseVariable-WalkSAT/Tabu;
```

Fig. 3. Outline of the function *chooseVariable* for WalkSAT/Tabu.

```
function chooseVariable-Novelty(s, Φ)
    c := randomPick({c | c clause of Φ and c not satisfied under s})
    L := List of variables occurring in c ordered according to decreasing
            score(x) = fix(Φ, s, x) − break(Φ, s, x) and age(x);
    v' = first element of list L;
    v'' = second element of list L;
    if age(v') > age(v'') then
        return v';
    else
        with probability wp: return v'';
        otherwise return v';
    end if
end chooseVariable-Novelty;
```

Fig. 4. Outline of the function *chooseVariable* for Novelty.

with the highest score is not the most recently flipped variable in the clause, it is flipped,[4] otherwise, with probability *wp* the second-best variable, and with probability $1 - wp$ the best variable is flipped. A pseudocode description of the variable selection strategy used in Novelty is given in Fig. 4. In analogy to [21] we will refer to the *wp* and *tl* parameters for the different heurisitics as *noise parameters*.

3. RTD-based empirical analysis of SLS algorithms

SLS algorithms like WalkSAT strongly involve random decisions such as the choice of the initial assignment, random tie-breaking, or biased random moves. Due to the inherent

[4] If several variables in the selected clause have identical score and age (which rarely happens, as clauses are typically rather short and the age of two variables can only be identical if both have never been flipped since the last search initialisation), ties are broken based on an arbitrary, fixed ordering of the variables appearing in the given formula.

randomness, the time needed by an SLS algorithm to find a solution differs from run to run even for a single instance. In general, the run-time needed by such an algorithm to find a solution is a random variable. Therefore most SLS algorithms are of *Las Vegas* type [1,20]; an algorithm A for a problem class Π is of Las Vegas type if

(i) for a given problem instance $\pi \in \Pi$ it returns a solution s, s is guaranteed to be a valid solution of π, and

(ii) on each given instance π, the run-time of A is a random variable $RT_{A,\pi}$.

For Las Vegas algorithms there are fundamentally different criteria for evaluation, depending on the characteristics of the environment they are supposed to work in. While in the case of no time limits being given (type 1 situation), the mean run-time might suffice to roughly characterise the run-time behaviour, in real-time situations with a given, strict time-limit (type 2 situation) it is basically meaningless. This is because the time-limits are often considerably lower than the expected or median run-time; at the same time, a Las Vegas algorithm with very high expected run-time might still give a reasonable solution probability for short runs. An adequate criterion for a type 2 situation with time-limit $t_{\max}$ is $P(RT \leqslant t_{\max})$, the probability of finding a solution within the given time-limit. In the most general case we are given a utility function that defines the usefulness or utility of a solution depending on the time needed to find it. In this case it is important to be able to characterise the run-time behaviour by the run-time distribution function rtd: $\mathbb{R} \mapsto [0, 1]$ defined as $rtd(t) = P(RT \leqslant t)$ or some approximation of it. This run-time distribution (RTD) completely and uniquely characterises the run-time behaviour of a Las Vegas algorithm. Given this information, other criteria, like the mean run-time, its standard deviation, percentiles, or success-probabilities $P(RT \leqslant t')$ for arbitrary time-limits t' can be easily obtained. [5]

To actually measure RTDs, one has to take into account that most SLS algorithms have some cutoff parameter bounding their run-time, like the *maxSteps* parameter in Fig. 1. Practically, we measure empirical RTDs by running the respective Las Vegas algorithm for n times (without using restart, i.e., setting *maxTries* to one) on a given problem instance up to some (very high) cutoff value [6] and recording for each successful run the time required to find a solution. The empirical run-time distribution is the cumulative distribution associated with these observations. More formally, let $rt(j)$ denote the run-time for the jth successful run, then the cumulative empirical RTD is defined by $\widehat{P}(rt \leqslant i) = |\{j \mid rt(j) \leqslant i\}|/n$. Instead of actually measuring run-*time* distributions in terms of CPU-time, it is often preferable to use representative operation counts as a more machine independent measure of an algorithm's performance. An appropriate operation count for local search algorithms for SAT, for example, is the number of local search steps, i.e., we get run-*length* distributions (RLDs) instead of run-*time* distributions. Note, that obtaining run-length distributions for single instances does not involve significantly higher computation times than to get a stable estimate for the mean performance of an algorithm.

Hooker [12,13] criticises that the empirical analysis of algorithms usually remains at the stage of simply collecting data and argues that, analogous to empirical methodology used in

[5] For a more general discussion of the application scenarios and, in general, of the advantages of measuring run-time distributions on single instances, we refer to [16].

[6] Optimal cutoff settings may then be determined *a posteriori* using the empirical run-time distribution.

other sciences, one should furthermore attempt to formulate hypotheses based on this data which, in turn, can be experimentally refuted or validated. Our RTD-based methodology, as outlined above, meets this demand: We first analyse the run-time behaviour of SLS algorithms for SAT on single, hard instances; then we use these observations as a basis for formulating hypotheses on the type of run-time distributions observed for these algorithms for various subclasses of SAT. We validate our hypotheses by testing them for a wide range of individual problem instances, using standard statistical methodology for testing our RLD characterisations for single problem instances. Note, that hypotheses concerning problem classes with an infinite number of instances (such as SAT) can generally be not experimentally verified and therefore often validation based on a finite number of individual instances is the best one can do.

4. Empirical results for Random-3-SAT

In this section, we first describe the generation of the Random-3-SAT test sets we used for our experiments; then we detail how we optimised WalkSAT's noise parameter for those problem instances, and finally we report our results regarding the characterisation of WalkSAT's behaviour on these test sets.

4.1. Generation of problem instances

The Random-3-SAT formulae we used for our experiments are propositional formulae in conjunctive normal form in which all clauses consist of exactly three literals. Given the number of variables n and clauses l, these formulae are randomly generated according to the *fixed clause length model* [22]: Each clause is produced by choosing three variables at random according to a uniform distribution (i.e., each variable is chosen independently of the others with a probability of $1/n$) and then negating each of these variables with probability 0.5. In the clause generation process, tautological clauses (i.e., clauses containing a literal and its negation) and clauses with multiple occurrences of the same literal are rejected; clauses are produced until l clauses (not considering rejected clauses) are generated.

Thus, for each value of n and l, a random distribution of Random-3-SAT formulae is given; test sets generated as described above correspond to samples from these distributions. It is well known that the probability for obtaining a satisfiable instance by sampling from a specific Random-3-SAT instance distribution critically depends on the clauses per variables ratio $r = n/l$. While for small r, most of the instances are underconstrained and thus satisfiable, at a certain critical value of r this changes rather abruptly. Beyond this critical r, most of the instances are overconstrained and therefore unsatisfiable. This so-called *phase-transition phenomenon* has been observed for a number of problem classes and received a lot of attention within the CSP and SAT communities [9]. For SAT, the critical r (the *phase-transition* or *cross-over point*), where approximately 50% of the randomly generated instances are soluble, occurs at $r \approx 4.3$ (for large n) [3, 22]; it is known that at this point the average hardness of the instances is maximal for both systematic and local search based SAT algorithms [28].

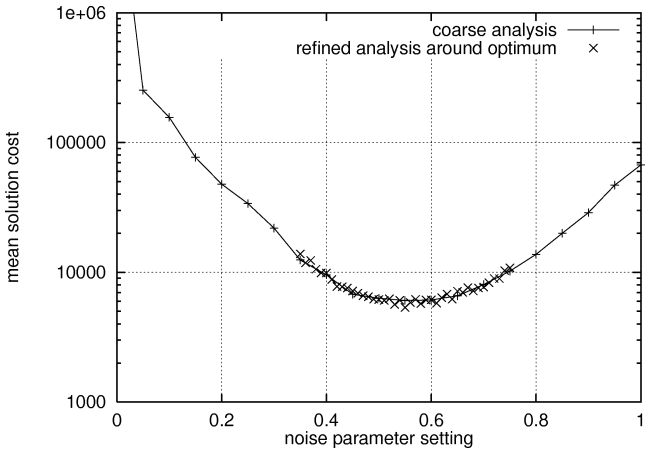

Fig. 5. Functional dependency of mean solution cost (expected number of variable flips for finding a solution) and noise parameter setting for a single Random-3-SAT instance with 100 variables, 430 clauses. The shape of the curve is typical for WalkSAT applied to hard Random-3-SAT instances (see text for details).

Test sets of Random-3-SAT formulae from the phase transition region are popular in SAT algorithm analysis, since they are easy to generate but hard to solve. However, when studying incomplete local search algorithms, there is no point in evaluating their performance on unsatisfiable instances; therefore we filtered the randomly generated instances using a fast complete SAT algorithm, such that the final test sets contain only satisfiable instances. This is well established methodology [21] and guarantees that if the tested algorithm fails to find a solution for a given instance, this is not caused by the instance being unsatisfiable.

4.2. Optimising the noise parameter

For the following empirical analyses, we generally use noise parameter (walk probability) settings which are chosen such that the expected number of steps required for finding a solution (solution cost) is minimal. Therefore, as a first step, we determined noise parameter settings for WalkSAT minimising the mean search cost when applied to individual problem instances. Since a precise determination of the optimal noise setting for each problem instance from our test sets is computationally not feasible, we used the following two-stage approach.

First, we randomly selected 10 problem instances from each test set and determined the mean local search cost for noise settings between 0.0 and 1.0 in increments of 0.05 from 1000 runs of the algorithm with $maxSteps = 100000$. This analysis strongly indicates that for a single instances the mean solution cost as a function of the noise setting is always convex with a single minimum located between 0.4 and 0.7; Fig. 5 shows a typical example for one of the 100 variable instances.

Next, we applied the same method, but using noise settings from 0.35 to 0.75 in 0.01 increments, to 100 randomly selected instances from the 50 and 100 variable test set and

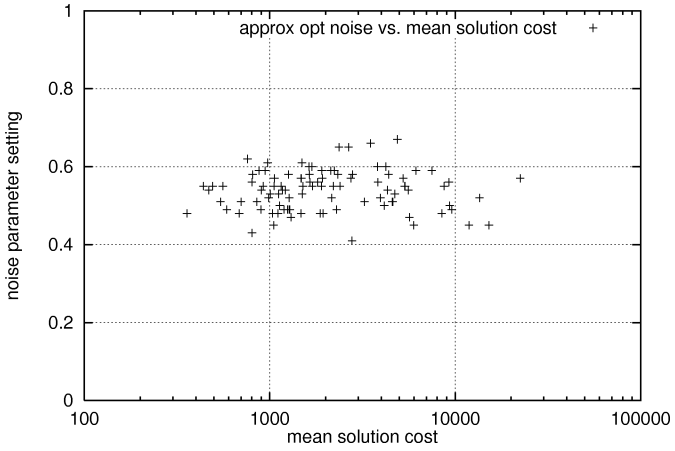

Fig. 6. Correlation between mean solution cost and noise parameter setting for 100 randomly generated Random-3-SAT instances with 100 variables, 430 clauses (see text for details).

Table 1
Basic statistics for hardness distributions

Test set	Mean	Median	Stddev	Stddev/mean
50 vars, 1000 inst	429.56	277	479.71	1.1168
100 vars, 1000 inst	2507.5	1433	3580.3	1.4278
200 vars, 100 inst	106440	10032	664100	6.2392

10 randomly selected instances from the 200 variable test set to verify this results and to determine the best noise settings with more accuracy. The results of this analysis show that the mean of the best noise settings over the samples is 0.57 for $n = 50$, 0.55 for $n = 100$, and 0.52 for $n = 200$. Although there is some variance in the optimal noise setting between the instances, this is relatively small and seems to decrease with problem size; the variation coefficients (= mean / standard deviation) are 0.14 for $n = 50$, 0.09 for $n = 100$ and 0.05 for $n = 200$. Furthermore, there is no correlation between the mean solution cost and the optimal noise setting within the test sets (absolute correlation coefficients < 0.05). To illustrate these results, in Fig. 6 we give a scatter plot of the mean solution cost and optimal noise settings for the 100 tested instances from the 100 variable test set; the corresponding results for $n = 50$ and $n = 200$ are analogous. It should also be noticed that around the optimal noise value the sensitivity of the search cost with respect to small changes of the noise setting is very low (see Fig. 5). Based on these results, we used an *approximately optimal* noise setting of 0.55 for WalkSAT for all subsequent experiments.

4.3. Distribution of the mean solution cost

To get an impression of the variability of the solution cost between the instances of the test sets, we first determined hardness distributions for test sets with 50, 100, and 200

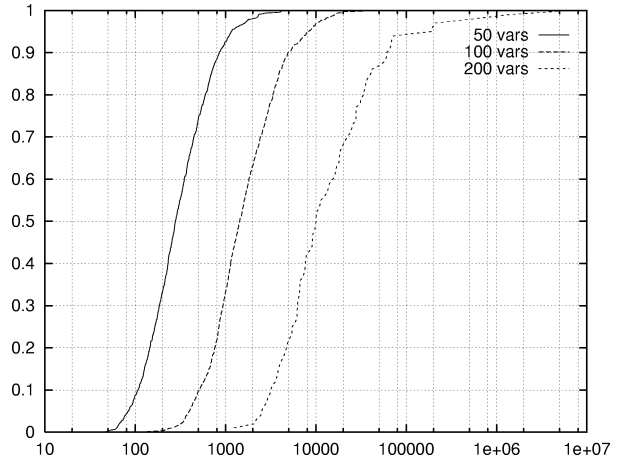

Fig. 7. Cumulative distributions of median local search cost (number of local search steps per solution) across Random-3-SAT test sets for different problems sizes. The x-axis shows the median search cost for WalkSAT, the y-axis shows the probability that the median search cost of a given instance is below a given bound (see text for details).

variables. Fig. 7 shows the cumulative distributions of the median search cost per instance (measured in local search steps) across the three test sets. The test sets for $n = 50, 100$ contain 1000 instances each, the one for $n = 200$ contains 100 instances. For each problem instance, the median search cost was determined from an RLD obtained by running WalkSAT with an approximately optimal noise parameter (determined as described above) of $wp = 0.55$ for 1000 tries. By using an extremely high cutoff parameter (i.e., allowing a very long time before the search is unsuccessfully aborted) we made sure that each instance was solved in every single try. In all our experiments with WalkSAT, using the noise parameter as specified above, we found that for sufficiently high cutoff parameter value, a solution was always found. This indicates that WalkSAT with the given noise parameter might be *probabilistically approximately complete*, i.e., for each soluble problem instance, the probability of solving it in a single try (without using restart) converges to 1 as the cutoff parameter (*maxSteps*) approaches infinity [15]. [7]

As can be seen from Fig. 7, there is a huge variability between the instances of each test set (see also Table 2). At the same time, for increasing problem sizes the tail of the distributions becomes more prominent, indicating that for larger problems there is considerably more variability in the median search cost, especially among the hardest instances from each test set. This can also be seen from the normalised standard deviations as given in Table 2. It is known that averaging over such extremely inhomogeneous test sets is potentially problematic [16]. Consequently, in the next step we analysed the RLD data for individual instances.

[7] Note that probabilistic approximate completeness is easy to prove if restart is used, but approximate completeness for the pure WalkSAT strategy, without restart, is yet unproven.

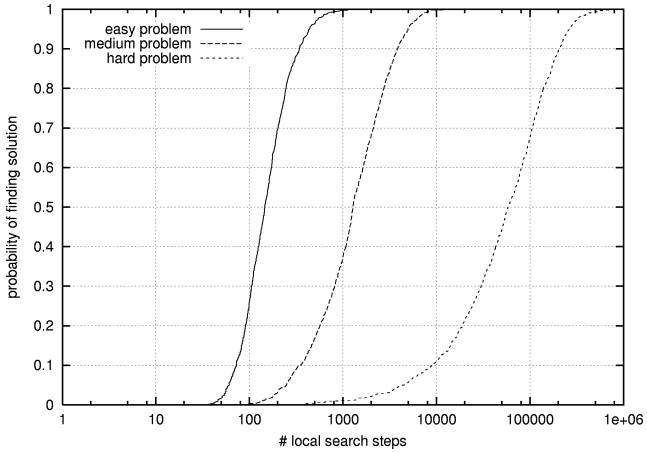

Fig. 8. RLDs for WalkSAT with $wp = 0.55$ and 1000 tries on the easiest, median, and hardest problem from the hardness distribution for the $n = 100$ test set.

Table 2
Parameter and quality of RLD approximation using exponential distributions $ed[m]$

Instance	Median #steps m	χ^2 for $ed[m]$
easy	132	501.68
medium	1433	69.41
hard	61082	27.51

4.4. Run-length distributions for individual instances

Fig. 8 shows the RLDs for WalkSAT (using the parameter settings as described before) when applied to the easiest, median, and hardest problem from the 100 variables, 430 clauses test set. For the hardest problem, the RLD can be approximated[8] using the cumulative form of an exponential distribution $ed[m](x) = 1 - 2^{-x/m}$, where m is the median of the distribution and x the number of steps required to find a solution.[9] For testing the goodness of this approximation we use a standard χ^2-test [25]. Basically, for a given empirical RLD this is done by estimating the parameter m and comparing the deviations to the predicted distribution $ed[m]$. The result of this comparison is the χ^2 value, where low χ^2 values indicate a close correspondence between empirical and predicted distribution. Table 2 shows the estimated parameters m and the χ^2 values for the easy, median, and hard instance mentioned before. It can be clearly seen that with increasing median search cost m the χ^2 value decreases, indicating that the harder the problem,

[8] All approximations were done using C. Gramme's "Gnufit" implementation of the Marquart–Levenberg algorithm for fitting parametric functions to a set of data points.

[9] In the statistical literature, the exponential distribution $Exp(\lambda)$ is usually defined by $P(X \leqslant x) = 1 - e^{-\lambda x}$, which is equivalent to our representation $ed[m]$ using $m = \ln 2/\lambda$.

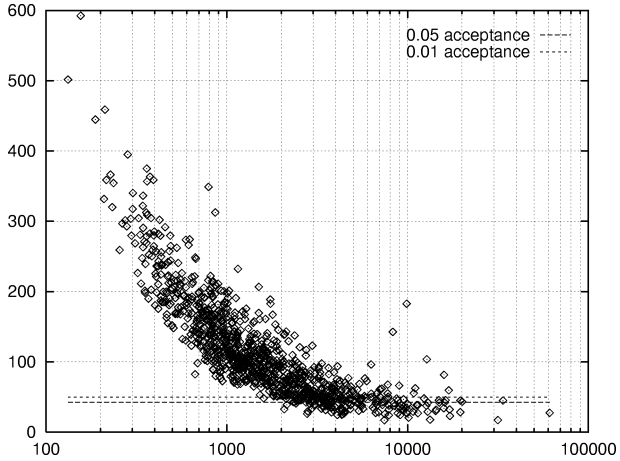

Fig. 9. Correlation between hardness of problems (x-axis, median number of local search steps per solution) and χ^2 values (y-axis) from testing the RLDs of individual instances versus a best-fit exponential distribution for $n = 100$ test set. The horizontal lines indicate the acceptance thresholds for the 0.01 and 0.05 acceptance level.

Table 3
Number of instances passing χ^2 test for different Random-3-SAT test sets

Test set	Acceptance level	Number passed
50 vars, 1000 inst	0.01	114 = 11.4%
50 vars, 1000 inst	0.05	71 = 7.1%
100 vars, 1000 inst	0.01	166 = 16.6%
100 vars, 1000 inst	0.05	103 = 10.3%
200 vars, 100 inst	0.01	30 = 30%
200 vars, 100 inst	0.05	26 = 26%

the closer WalkSAT's RLD on this problem approximates an exponential distribution. In the given case, to pass the χ^2 test at a standard $\alpha = 0.05$ acceptance level, a $\chi^2 \leqslant 49.6$ is required. Thus, only the approximation for the hard instance passes the test. Note, however, that for the median and easiest problem, the approximation is still reasonably good for the tail of the distribution (i.e., for long runs), while for smaller number of steps the actual distribution is steeper than an exponential distribution. We conjecture that the deviations are caused by the initial hill-climb phase of the local search procedure (cf. [6]): WalkSAT (and all other algorithms used in our study) starts its search from a randomly chosen assignment, which typically violates many clauses. Consequently, the algorithm needs some time to reach the first local optimum (which possibly could be a satisfying solution); in this initial phase of the search the probability for finding a solution is zero.

These observations lead to the following hypothesis: *For hard Random-3-SAT instances from the phase transition region, the run-time behaviour of WalkSAT with approximately*

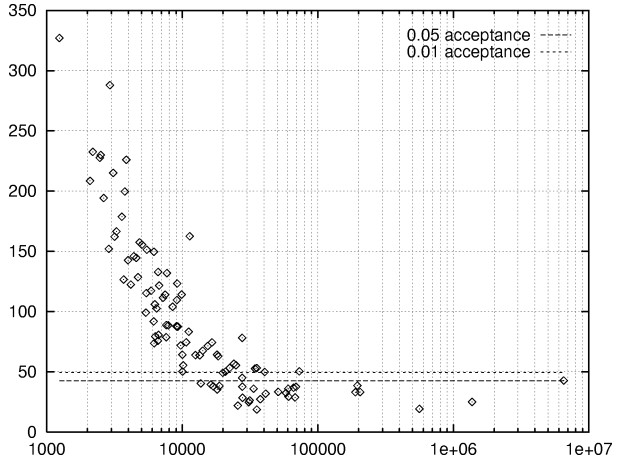

Fig. 10. Correlation between hardness of problems (x-axis, median number of local search steps per solution) and χ^2 values (y-axis) from testing the RLDs of individual instances versus a best-fit exponential distribution for $n = 200$ test set. The horizontal lines indicate the acceptance thresholds for the 0.01 and 0.05 acceptance level.

Table 4
Parameter and quality of RLD approximation using exponential distributions $ed[m]$

Instance	Median #steps m	χ^2 for $ed[m]$
bw_large.a	6839	60.37
bw_large.b	119680	11.40 [a]
bw_large.c	4.27×10^6	8.71 [a]

[a] Passed the χ^2-test at the $\alpha = 0.05$ acceptance level.

optimal noise setting can be characterised using exponential distributions. To further investigate this hypothesis, we apply the methodology outlined above to the entire test sets. The resulting correlation between median search cost and χ^2 values can be seen from the scatter plots given in Figs. 9 and 10. Obviously, there is a strong negative correlation, indicating that, indeed, for harder problem instances, the χ^2 values tend to be lower which leads to the conclusion that WalkSAT's behaviour can be more and more accurately characterised by exponential distributions. The figures also indicate two standard acceptance levels for the χ^2 test ($\alpha = 0.01$ and $\alpha = 0.05$). As can be seen from the plots, for high median search cost, almost all instances pass the χ^2 test. Table 3 shows the overall percentage of the instances which passed the test for the different acceptance levels. The data suggests that for increasing problem size, a relatively higher number of instances pass the test, i.e., the deviations of the RLDs from ideal exponential distributions become less prominent for larger problems. [10]

[10] The complete RLD data for the experiments described here is available from the authors.

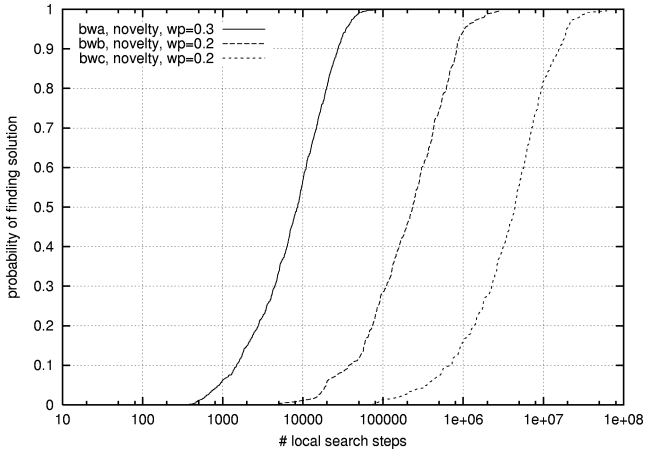

Fig. 11. RLDs for Novelty with approx. optimal noise setting on SAT encoded blocks world planning problems.

In summary, the data from the χ^2 tests confirms and refines our hypothesis that for hard Random-3-SAT instances from the phase transition region, the run-time behaviour of WalkSAT with approximately optimal noise setting can be approximated using exponential distributions.

5. Beyond random-3-SAT

Applying the methodology described in the previous sections to other SLS algorithms and SAT-encoded problems from other domains we obtain similar results as described for the random problem distributions.

Fig. 11 shows the RLDs for Novelty, one of the most recent WalkSAT variants, when applied to SAT encoded problem instances from the blocks world planning domain.[11] We used approximately optimal noise parameter settings ($wp = 0.4$, 0.3, 0.2 for the blocks world planning instances `bw_large.a,b,c`), determined similarily as described in Section 4.2, and *maxSteps* settings which were high enough (10^7 and 10^8, respectively) to ensure 100% success rate. For `bw_large.a` we used 1000 runs, for the other problems 250 runs to approximate the actual RLDs as described before. The estimates for the parameter m and the corresponding χ^2 values for the approximation by exponential distributions are shown in Table 4. The critical χ^2 values for a standard $\alpha = 0.05$ acceptance level are 42.6 for 1000 tries and 26.3 for 250 tries. This means, that only for the smallest instance the approximation does not pass the test. As for the easy Random-3-SAT instances, we interpret this as an effect of the initial hill-climb phase of local search.

[11] We used the problem instances from [19] in the SAT formulation obtained by linear encoding and simplifying. Similar results were obtained for other problem instances from the same domain.

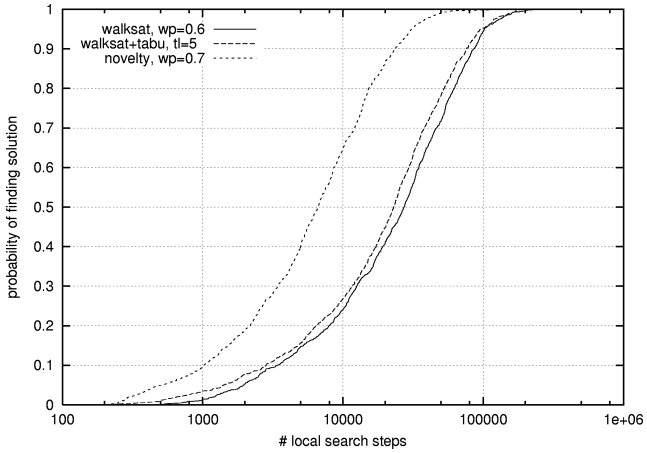

Fig. 12. RLDs for WalkSAT and WalkSAT/Tabu, and Novelty with approx. optimal noise settings on SAT encoded graph colouring problem.

Table 5
Parameter and quality of RLD approximation using exponential distributions $ed[m]$

Algorithm	Median #steps m	χ^2 for $ed[m]$
WalkSAT	26267	41.92
WalkSAT/Tabu	22450	26.26
Novelty	6722	42.37

In a final experiment, we applied the same methodology to a SAT-encoded problem instance [12] from the graph colouring domain, using three different variants of WalkSAT (standard WalkSAT, WalkSAT with tabu-search, and Novelty [21]). Again, we determined approximately optimal noise parameter settings as described in Section 4.2 ($wp = 0.6$ for WalkSAT, a tabu-list length of 5 for WalkSAT/Tabu, and $wp = 0.7$ for Novelty). For each algorithm we performed 1000 tries, using a large *maxSteps* value (10^6). The experimental results are shown in Fig. 12 and Table 5. For all three algorithms, the approximations of the empirical RLDs using exponential distributions passed the standard χ^2 test for $\alpha = 0.05$.

In summary, the results presented in this section show that the hypothesis which we formulated and tested for Random-3-SAT in the previous section also holds for hard SAT-encoded problem instances from other domains. Furthermore, it is not restricted to WalkSAT, but also applies to some variants of the basic algorithm which show significantly better performance. Based on our experimental experience we conjecture that the regular behaviour we observed for these algorithms when applied to hard problem instances might be a general property of current SLS algorithms for SAT.

[12] A 3-colourable instance with 75 vertices and connectivity 4.6 (corresponding to the phase transition for this type of graph colouring problem [11]).

6. Interpretation

The empirical results presented in Sections 3 and 4 have a number of theoretically and practically interesting consequences.

Random restart

For algorithms exhibiting an exponential RLD, the probability of finding a solution within a fixed time interval is independent of the run-time spent before. Based on our results, this holds for various current SLS algorithms for SAT when using approximately optimal noise parameter settings. Further experimentation indicates that exponential RLDs are also characteristic for noise parameter values which are larger than the optimal noise parameter settings (cf. [14]). For example, when analysing WalkSAT's behaviour for the $n = 100$ Random-3-SAT test-set as outlined in Section 3, but using larger-than-optimal noise settings of $wp = 0.65$ and 0.75, we get acceptance rates of 19.8% and 29.0%, respectively, from the χ^2 test with acceptance level $\alpha = 0.05$. [13]

Thus, for approximately optimal and larger-than-optimal noise settings, these SLS algorithms are essentially memoryless, as for a given total time t, restarting at time $t' < t$ does not significantly influence the probability of finding a solution in time t. [14] Consequently, for these algorithms random restart is ineffective. In practice this result can be easily verified over a broad range of *maxSteps* settings. However, due to small deviations at the extreme left end of the RLDs, for very low *maxSteps* settings the algorithms' performance usually deteriorates. As argued in Section 3, this is most likely an effect of the initial hill-climbing phase of SLS algorithms based on hill-climbing approaches.

For smaller-than-optimal settings of the noise parameter, a different situation can be found (cf. [14]). The RLDs are less steep than exponential distributions, which means that the efficiency of the search process decreases over time and therefore random restart can be effectively used to speed up the algorithm. A typical example for this is given in Fig. 13, showing the RLDs for Novelty applied to the hardest instance from the $n = 100$ Random-3-SAT test set using approximately optimal and smaller-than-optimal noise settings. Note that for small runs the otherwise inferior small noise settings achieve higher success probabilities. Therefore, using, e.g., $wp = 0.4$ and restarting after $maxStep = 500$ steps in this particular case gives even better performance than using $wp = 0.7$ with any *maxSteps* setting. However, for $wp = 0.4$, Novelty's performance is very sensitive with respect to *maxSteps*, while for $wp = 0.7$, the RLD is very well approximated by an exponential distribution (the standard χ^2-test accepts with a χ^2 value of 33.79) and thus, as argued above, the algorithm's performance is essentially independent of *maxSteps*. This example demonstrates the usefulness of the RLD-based analysis: the tradeoff between performance and robustness (with respect to restart) can be easily seen from the RLDs in Fig. 13, while comparing means, or even the standard percentiles (0.1,0.25,median, 0.75,0.9) would not have revealed this phenomenon.

[13] As reported in Table 4, for the approximately optimal noise value of $wp = 0.55$, the corresponding acceptance rate is 16.6%.

[14] Because the exponential distribution is memoryless, it is often used in reliability theory to describe components that are not subject to aging phenomena, like transistors [2].

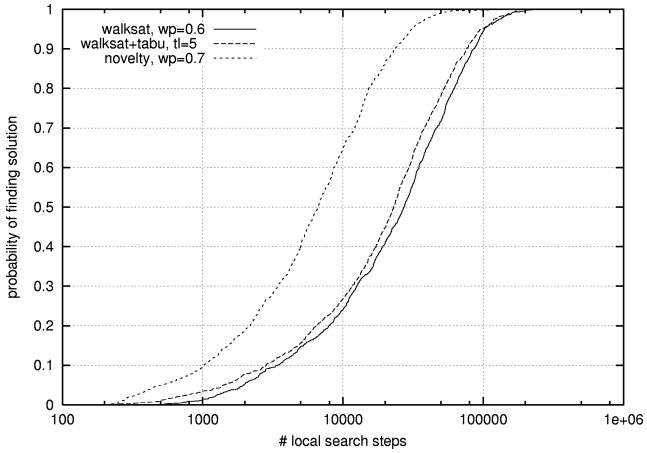

Fig. 13. RLDs for Novelty applied to the hardest instance from the $n = 100$ Random-3-SAT test set from Section 3, using approximately optimal ($wp = 0.7$) and smaller-than-optimal noise settings.

Consequences for parallel processing

Randomised algorithms lend themselves to a straightforward parallelisation approach by performing independent runs of the same algorithm in parallel. Given our characterisations of exponential RLDs for SAT, this approach is particularly effective: Based on a well-known result from statistical literature [25], if for a given algorithm the probability of finding a solution in t time units is distributed exponentially $ed[m]$, then the probability of finding a solution in k independent runs of time t is distributed $ed[m/k]$. Consequently, if we run such an algorithm once for time t we get the same success probability as when running the algorithm k times for time t/k. This means that using multiple independent runs of the corresponding local search algorithms, an optimal speedup can be achieved. This holds for almost arbitrary numbers k of processors; only for high k, due to the deviations for short runs caused by the initial hill-climb phase of local search, the speedup would be less than optimal. Note that again this result holds for optimal as well as larger-than-optimal noise parameter settings.

For smaller-than-optimal settings, if the overall processing time t (number of processors $\times$ time per processor) is fixed, a single optimal number of processors can be derived from the RTD data, for which a super-optimal speedup can be obtained when compared to the sequential case $k = 1$. Formally, for a given problem instance let $p_s(1, t)$ be the success probability achieved by using a single processor and total processing time t. Then the success probability when running k independent runs on different processors in parallel with processing time t/k each is $p_s(k, t/k) = (1 - (1 - p_s(1, t/k))^k)$. Consequently, for a given RTD the optimal number of processors is given by the k' maximising $p_s(k', t/k')$. It can be easily verified that this k' also achieves the maximal speedup.

It should be noted that parallel execution of independent tries is an extremely attractive model of parallel processing, since it involves basically no communication overhead and can be easily implemented and run on almost any parallel hardware platform, from

networks of standard workstations to specialised MIMD machines with thousands of processors. Therefore, these results are not only relevant for the application of SLS algorithms to hard combinatorial problems like SAT to time-critical tasks (like robot control or online scheduling), but also for the distributed solving of very large and hard problem instances.

7. Related work and conclusions

In this paper we have characterised run-time distributions for WalkSAT, a high performing stochastic local search algorithms for SAT. We have shown that for approximately optimal noise parameter settings the RTDs for single hard Random 3-SAT instances and for SAT encoded problems can be approximated by exponential distributions. Compared to using simple descriptive statistics, as usually done in previous work, analysing run length distribution gives a more detailed information of SLS algorithms' behaviour and performance while not causing significant computational overhead. Additionally, RLDs provide the information required for assessing SLS algorithms in the context of more general application scenarios, such as realtime or anytime situations, where often the runtime limitations imposed by the application environment are so strict that the expected or median run-time is often irrelevant (cf. Section 2).

To the best of our knowledge, run-time distributions have not yet been investigated for local search algorithms on the SAT domain. In [4], the search cost distribution of finding a satisfying assignment on an *ensemble* of instances for *complete* procedures for SAT is studied. The distribution of the search cost for SLS algorithms corresponds to the hardness distribution shown in Fig. 7. For the hardness distributions we could not find a good approximation using any of the standard families of probability distributions. This is mainly due to the long tail on the right side of the empirical hardness distributions representing the hardest instances.

To clearly separate between different sources of the high variability observed in SLS algorithms' run-time behaviour when applied to test sets of randomly generated problem instances, our methodology is based on empirically studying SLS behaviour on single instances. A similar approach also proved to be essential for assessing the potential of parallel processing for *randomised complete* backtracking algorithms for graph colouring [10].[15] This shows that focussing on observations for single problem instances can be also very useful for the analysis of randomised complete search algorithms. Along the same lines, [7] observe distributions for single instances and find heavy-tailed distributions of the search cost for randomised complete procedures on the *Quasigroup Completion Problem*. For the SLS algorithms studied here, on the contrary, we could observe such heavy-tailed distributions neither as RLDs nor as hardness distributions.

Our work demonstrates how a more adequate methodology of empirically analysing the run-time behaviour of SLS algorithms can be used to obtain new and surprisingly general hypotheses. These hypotheses (concerning the type of RLDs and its dependency on the

[15] These algorithms are complete but involve randomised decisions in the single nodes of the search tree.

noise parameter setting, one of the crucial parameters of many current SLS algorithms) can be experimentally verified using standard statistical testing procedures. One such hypothesis that we could validate is that the RLDs on hard random 3-SAT instances from the phase transition region and SAT-encoded problems from other domains can be characterised by exponential distributions. Generalising the individual instance behaviour on a whole problem class is also a main motivation for approaches like the MULTITAC system [23] which automatically configures search algorithms for a given problem class based on a training set of typical instances.

From the results of our empirical investigation we also derived a number of both theoretically and practically interesting consequences. Moreover, our results give rise to a number of new questions:

- What are the common features of the different SLS approaches leading to the uniform behaviour (i.e., the exponential run-time distributions) for hard instances from various problem classes reported here?
- What causes the observed run-time behaviour? Exponential distributions are characteristic for the simplest randomised search technique, uniform random picking from a set of solution candidates. Thus, observing this type of behaviour for more sophisticated and much more efficient algorithms (such as the SLS algorithms studied here) suggests that their behaviour might be interpreted as random sampling from a much smaller space. Evidence for this interpretation could be established by linking this hypothetical "virtual searchspace" to features of the actual search space, such as the number, size, and topology of local optima.
- How can these results be generalised to optimisation problems? Since most SLS algorithms for decision problems like SAT implicitly solve the corresponding optimisation problem (here: MAX-SAT, because solutions for SAT are found by stochastically maximising the number of satisfied clauses), this generalisation is very natural and should provide further insight in the behaviour of SLS algorithms.

We are convinced that pursuing this line of research will significantly contribute to deepening our understanding of stochastic local search—and thus facilitate the development and application of improved SLS techniques.

Acknowledgement

We would like to thank Bart Selman, David McAllester, David Poole, Wolfgang Bibel, and the members of the Intellectics Group at TU Darmstadt for interesting discussions and comments on earlier versions of this article. We gratefully acknowledge the comments and suggestions of the anonymous reviewers, which greatly helped to improve the presentation of our work. H.H. acknowledges support by the German National Merit Foundation and IRIS-III Project "Interactive Optimization and Preference Elicitation" (BOU). T.S. acknowledges support by the Deutsche Forschungsgemeinschaft (DFG) via the Graduiertenkolleg "Intelligente Systeme für die informations- und Automatisierungstechnik" and a Marie Curie Fellowship (CEC-TMR Contract No. ERB4001GT973400).

References

[1] H. Alt, L. Guibas, K. Mehlhorn, R. Karp, A. Wigderson, A method for obtaining randomized algorithms with small tail probabilities, Algorithmica 16 (1996) 543–547.

[2] R.E. Barlow, F. Proschan, Statistical Theory of Reliability and Life Testing, Holt, Reinhart and Winston, 1981 (Reprint).

[3] J.M. Crawford, L.D. Auton, Experimental results on the crossover point in Random-3-SAT, Artificial Intelligence 81 (1–2) (1996) 31–57.

[4] D. Frost, I. Rish, L. Vila, Summarizing CSP hardness with continuous probability distributions, in: Proc. AAAI-97, Providence, RI, 1997, pp. 327–333.

[5] I.P. Gent, T. Walsh, Towards an understanding of hill-climbing procedures for SAT, in: Proc. AAAI-93, Washington, DC, 1993, pp.18–33.

[6] I.P. Gent, T. Walsh, An empirical analysis of search in GSAT, Washington, DC, J. Artificial Intelligence Res. 1 (1993) 47–59.

[7] C.P. Gomes, B. Selman, N. Crato, Heavy-tailed distributions in combinatorial search, in: Proc. CP-97, Lecture Notes in Computer Science, Vol. 1330, Springer, Berlin, 1997, pp. 121–135.

[8] J. Gu, Efficient local search for very large-scale satisfiability problems, SIGART Bulletin 3 (1992) 8–12.

[9] T. Hogg, B.A. Huberman, C.P. Williams, Phase transitions and the search problem (Editorial), Artificial Intelligence 81 (1–2) (1996) 1–16.

[10] T. Hogg, C.P. Williams, Expected gains from parallelizing constraint solving for hard problems, in: Proc. AAAI-94, Seatle, WA, 1994, pp. 331–336.

[11] T. Hogg, Refining the phase transition in combinatorial search, Artificial Intelligence 81 (1996) 127–154.

[12] J.N. Hooker, Needed: An empirical science of algorithms, Oper. Res. 42 (2) (1994) 201–212.

[13] J.N. Hooker, Testing heuristics: We have it all wrong, J. Heuristics (1996) 33–42.

[14] H.H. Hoos, Stochastic local search—Methods, models, applications, Ph.D. Thesis, TU Darmstadt, Germany, 1998. Electronically available from `http://www.cs.ubc.ca/spider/hoos/phd-thesis.html`.

[15] H.H. Hoos, On the run-time behaviour of stochastic local search algorithms for SAT, in: Proc. AAAI-99, Orlando, FL, 1999. Electronically available from `http://www.cs.ubc.ca/spider/hoos/publ-ai.html`.

[16] H.H. Hoos, T. Stützle, Evaluating Las Vegas algorithms—Pitfalls and remedies, in: Proc. UAI-98, Morgan Kaufmann, San Mateo, CA, 1998, 238–245.

[17] H.H. Hoos, T. Stützle, Characterizing the run-time behavior of stochastic local search, Technical Report AIDA-98-1, FG Intellektik, TU Darmstadt, 1998.

[18] Y. Jiang, H. Kautz, B. Selman, Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT, in: Proc. 1st Workshop on Artificial Intelligence and Operations Research, 1995.

[19] H. Kautz, B. Selman, Pushing the envelope: Planning, propositional logic, and stochastic search, in: Proc. AAAI-96, Vol. 2, Portland, OR, MIT Press, Cambridge, MA, 1996, pp. 1194–1201.

[20] M. Luby, A. Sinclair, D. Zuckerman, Optimal speedup of Las Vegas algorithms, Inform. Process. Lett. 47 (1993) 173–180.

[21] D. McAllester, B. Selman, H. Kautz, Evidence for invariants in local search, in: Proc. AAAI-97, Providence, RI, 1997, pp. 321–326.

[22] D. Mitchell, B. Selman, H. Levesque, Hard and easy distributions of SAT problems, in: Proc. AAAI-92, San Jose, CA, 1992, pp. 459–465.

[23] S. Minton, Automatically configuring constraint satisfaction programs: A case study, Constraints 1 (1) 1996.

[24] P. Morris, The breakout method for escaping from local minima, in: Proc. AAAI-93, Washington, DC, 1993, pp. 40–45.

[25] V.K. Rohatgi, An Introduction to Probability Theory and Mathematical Statistics, Wiley, New York, 1976.

[26] B. Selman, H.A. Kautz, B. Cohen, Noise strategies for improving local search, in: Proc. AAAI-94, Seatle, WA, MIT Press, Cambridge, MA, 1994, pp. 337–343.

[27] B. Selman, H. Levesque, D. Mitchell, A new method for solving hard satisfiability problems, in: Proc. AAAI-92, San Jose, CA, MIT Press, Cambridge, MA, 1992, pp. 440–446.

[28] T. Yokoo, Why adding more constraints makes a problem easier for hill-climbing algorithms: Analyzing landscapes of CSPs, in: Proc. CP-97, Springer, Berlin, 1997, pp. 357–370.